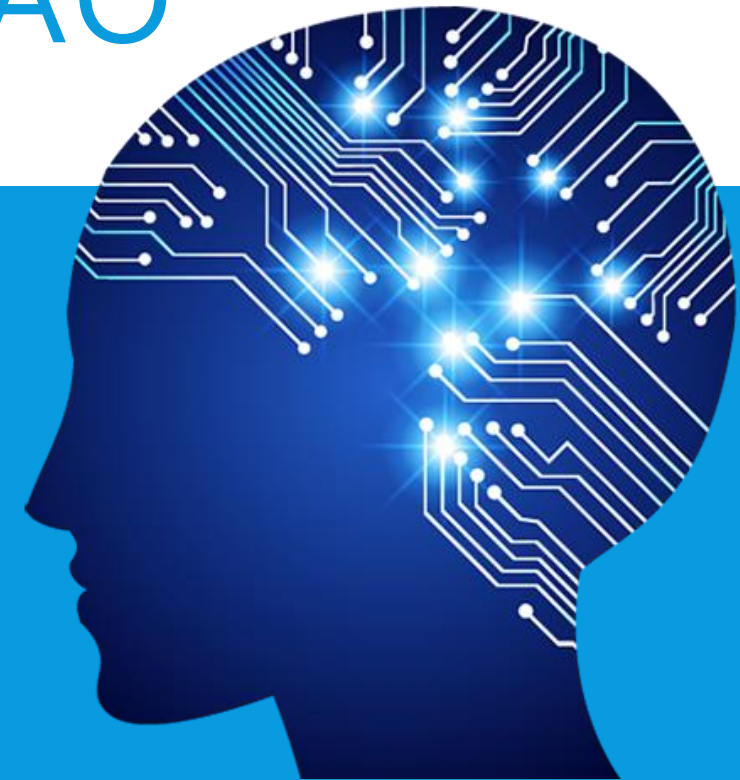


LÓGICA DE PROGRAMAÇÃO

Faculdade de Tecnologia Senac Pelotas

Escola de Tecnologia da Informação

Prof. Edécio Fernando Iepsen



ROTEIRO INICIAL DE UM PROGRAMA:

Entrada, processamento e saída

- a) Leia os dados de *entrada*.
- b) Realize o *processamento* dos dados.
- c) Apresente a *saída* dos dados.

CONCEITOS BÁSICOS

- Um programa é construído a partir de um conjunto de comandos, salvos em um arquivo.
- O nome do arquivo, precisa possuir uma extensão relacionada com a Linguagem de Programação escolhida (.js, .py, .ts, .php).
- Após, deve-se executar o programa, a partir de uma instrução (geralmente em linha de comando).

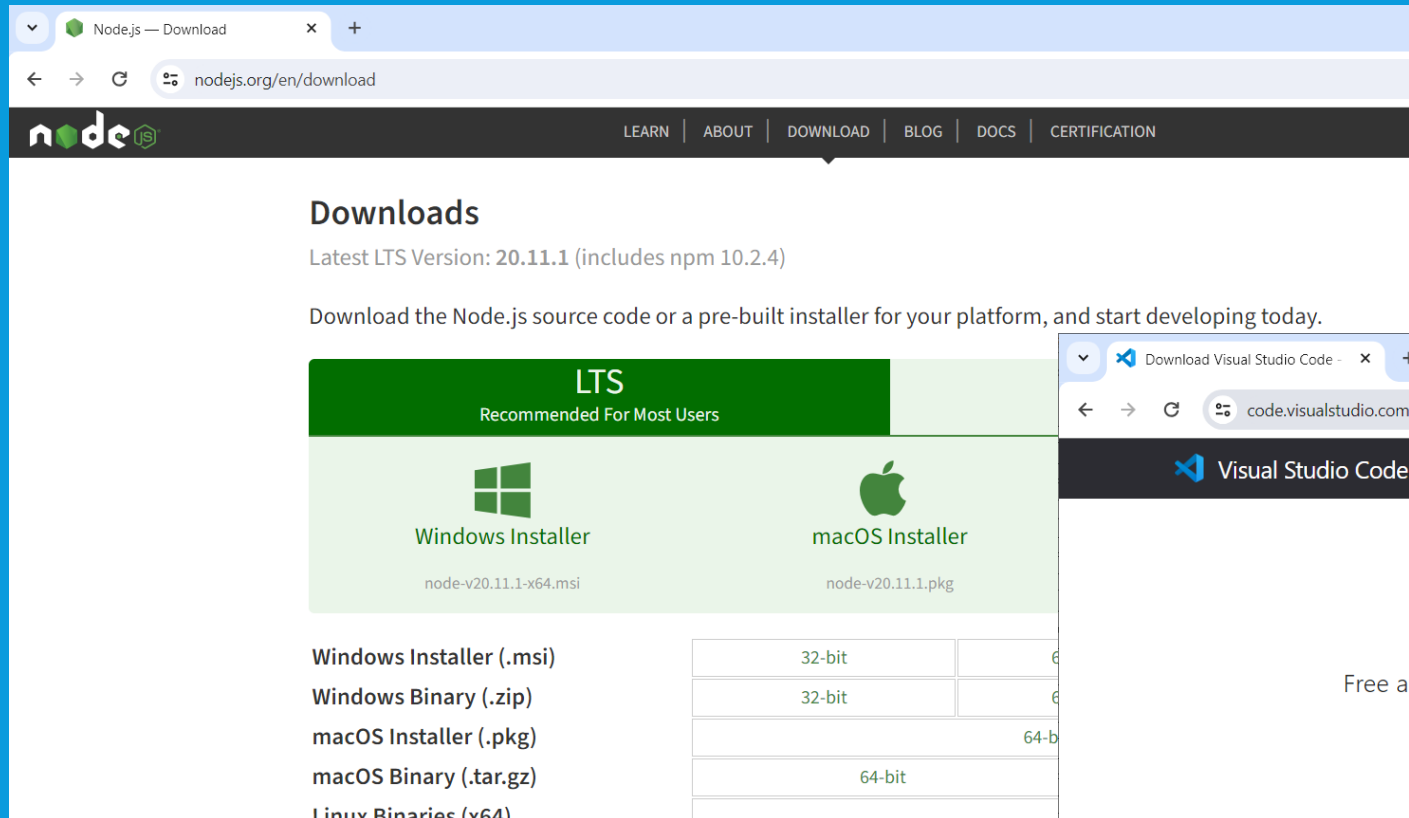
COMANDOS DE ENTRADA E SAÍDA

- Comandos JavaScript:

`console.log("Texto")` => saída de dados

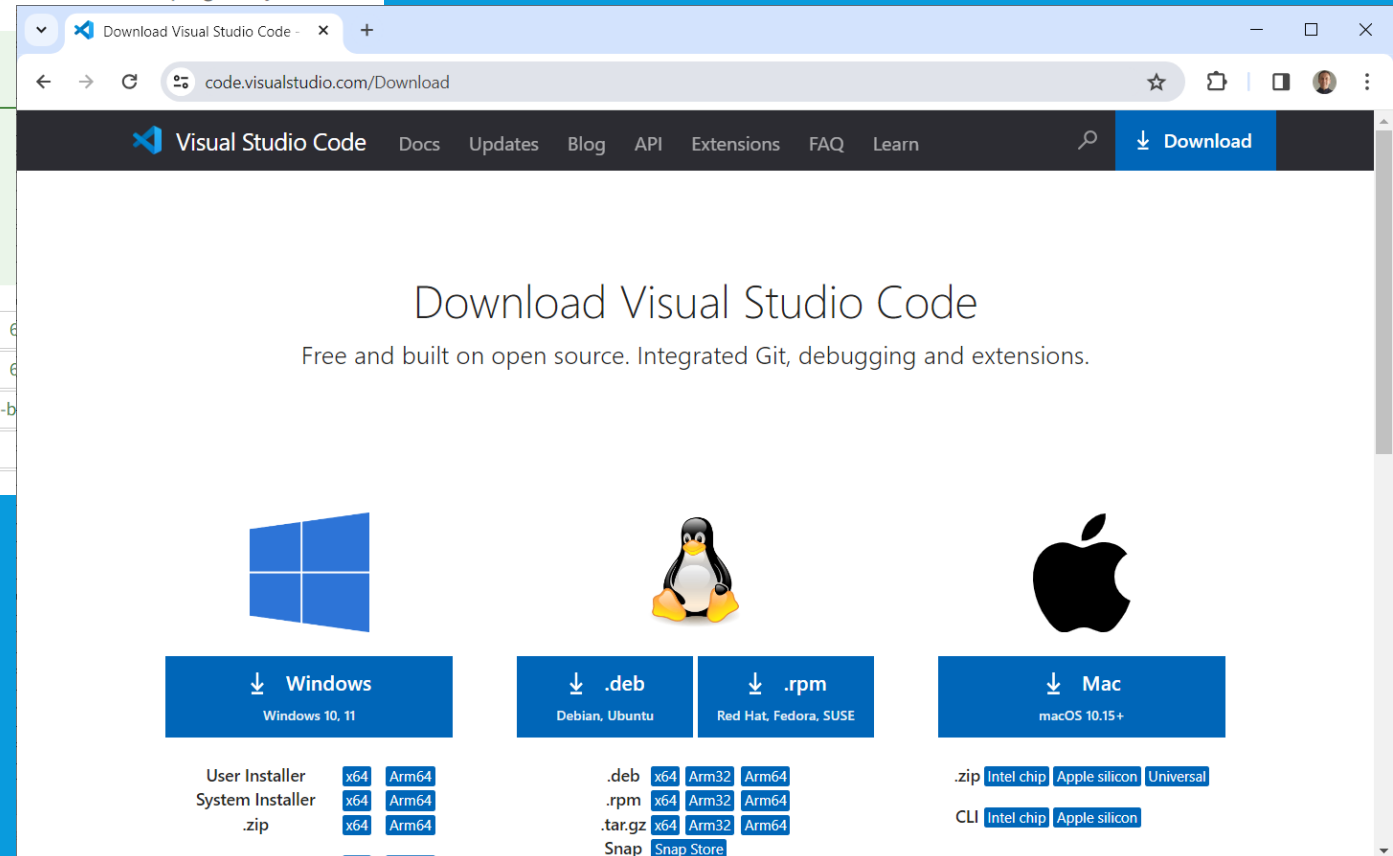
`prompt("Pergunta: ")` => entrada de dados

Softwares utilizados: Nodejs e Visual Studio Code



The screenshot shows the Node.js download page. The browser tab is 'Node.js — Download' and the address bar shows 'nodejs.org/en/download'. The page has a dark header with the Node.js logo and navigation links: LEARN, ABOUT, DOWNLOAD, BLOG, DOCS, and CERTIFICATION. The main content area is titled 'Downloads' and states 'Latest LTS Version: 20.11.1 (includes npm 10.2.4)'. It encourages users to download the Node.js source code or a pre-built installer. A green banner highlights 'LTS Recommended For Most Users'. Below this, there are two main download options: 'Windows Installer' (node-v20.11.1-x64.msi) and 'macOS Installer' (node-v20.11.1.pkg). A table lists additional download options: Windows Installer (.msi), Windows Binary (.zip), macOS Installer (.pkg), macOS Binary (.tar.gz), and Linux Binaries (x64).

Platform	Architecture	File Name
Windows	32-bit	node-v20.11.1-x64.msi
Windows	32-bit	node-v20.11.1-x64.msi
macOS	64-bit	node-v20.11.1.pkg
macOS	64-bit	node-v20.11.1.pkg
Linux	64-bit	node-v20.11.1-x64.msi

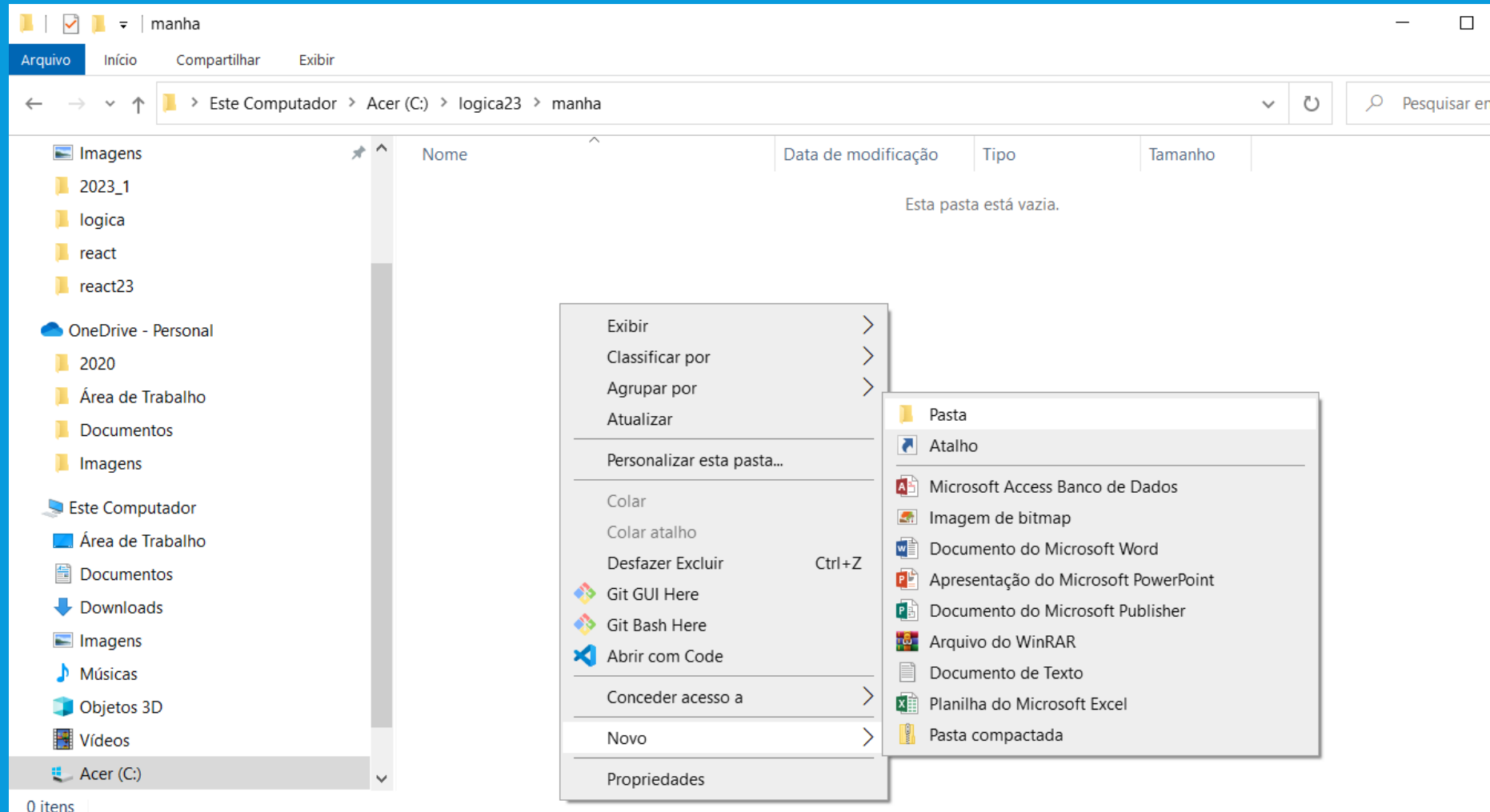


The screenshot shows the Visual Studio Code download page. The browser tab is 'Download Visual Studio Code' and the address bar shows 'code.visualstudio.com/Download'. The page has a dark header with the Visual Studio Code logo and navigation links: Docs, Updates, Blog, API, Extensions, FAQ, Learn, and a 'Download' button. The main content area is titled 'Download Visual Studio Code' and states 'Free and built on open source. Integrated Git, debugging and extensions.' Below this, there are three main download options: 'Windows' (Windows 10, 11), '.deb' (Debian, Ubuntu), and '.rpm' (Red Hat, Fedora, SUSE). A table lists additional download options: User Installer, System Installer, .zip, .deb, .rpm, .tar.gz, and CLI.

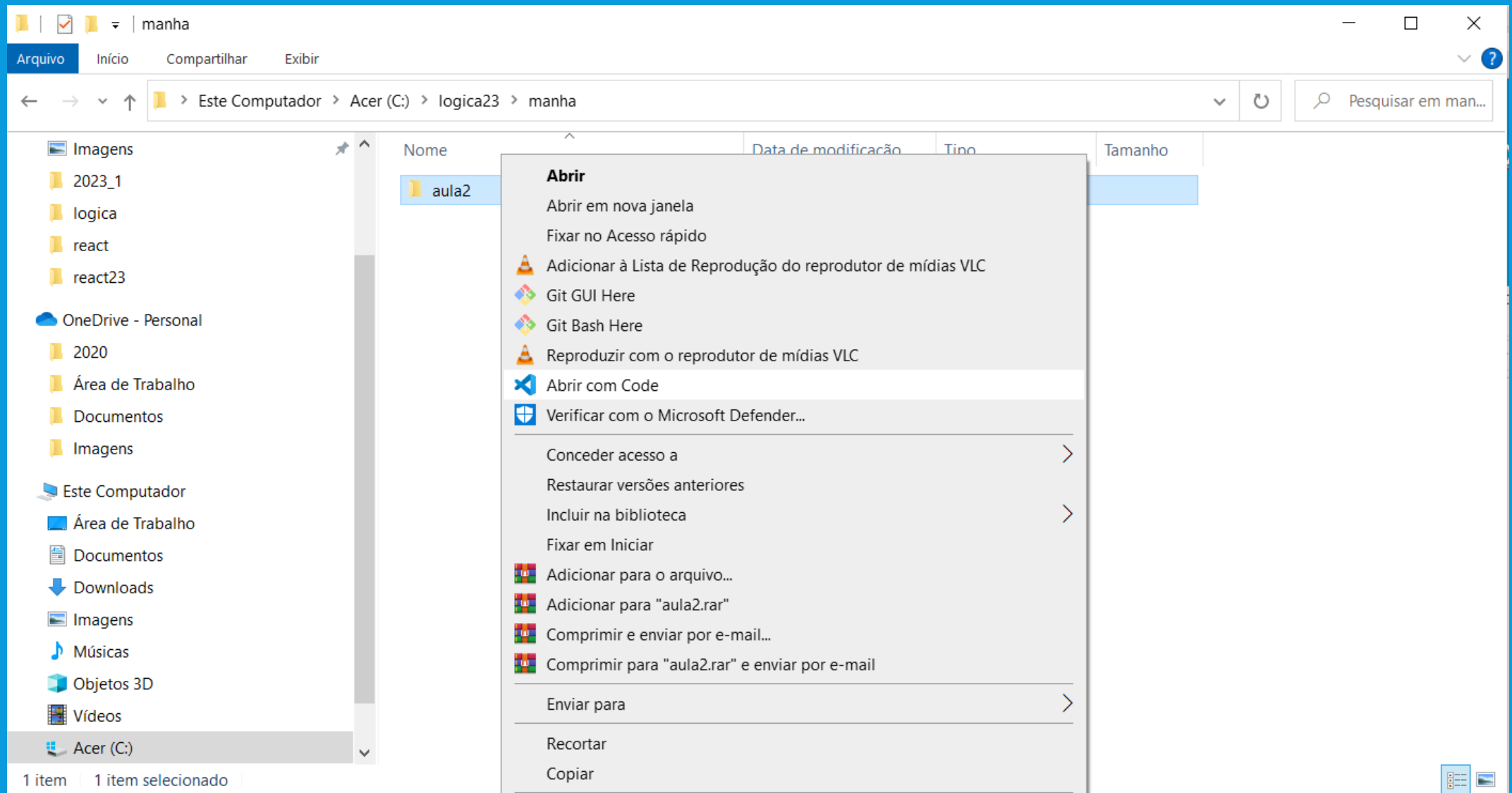
Platform	Architecture	File Name
Windows	Windows 10, 11	Windows 10, 11
.deb	Debian, Ubuntu	Debian, Ubuntu
.rpm	Red Hat, Fedora, SUSE	Red Hat, Fedora, SUSE
Mac	macOS 10.15+	macOS 10.15+

Passos para executar os programas com Node (JavaScript)

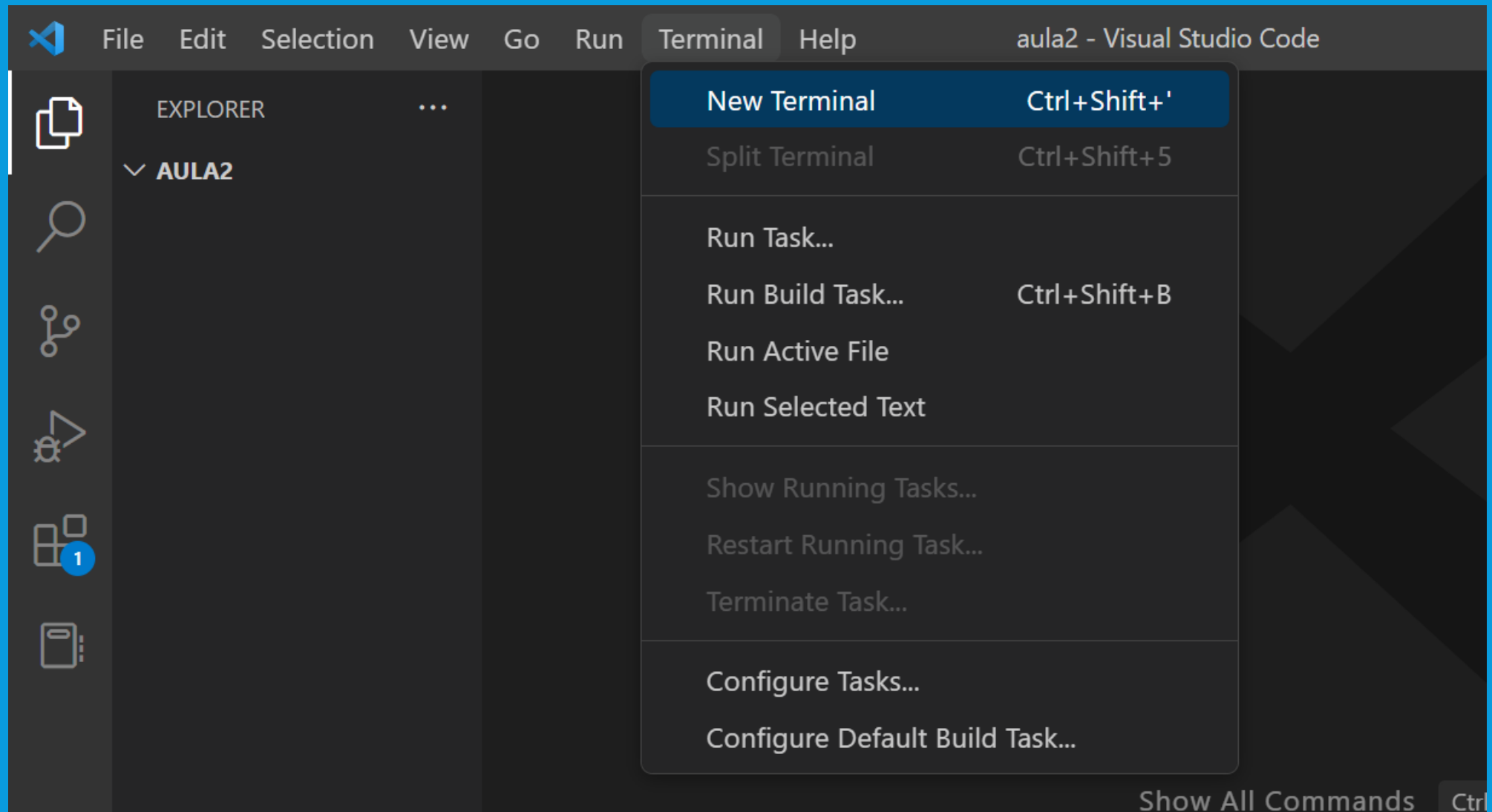
1. Criar uma pasta para organizar os programas da aula



2. Botão direito do mouse sobre a pasta e selecione Abrir com Code

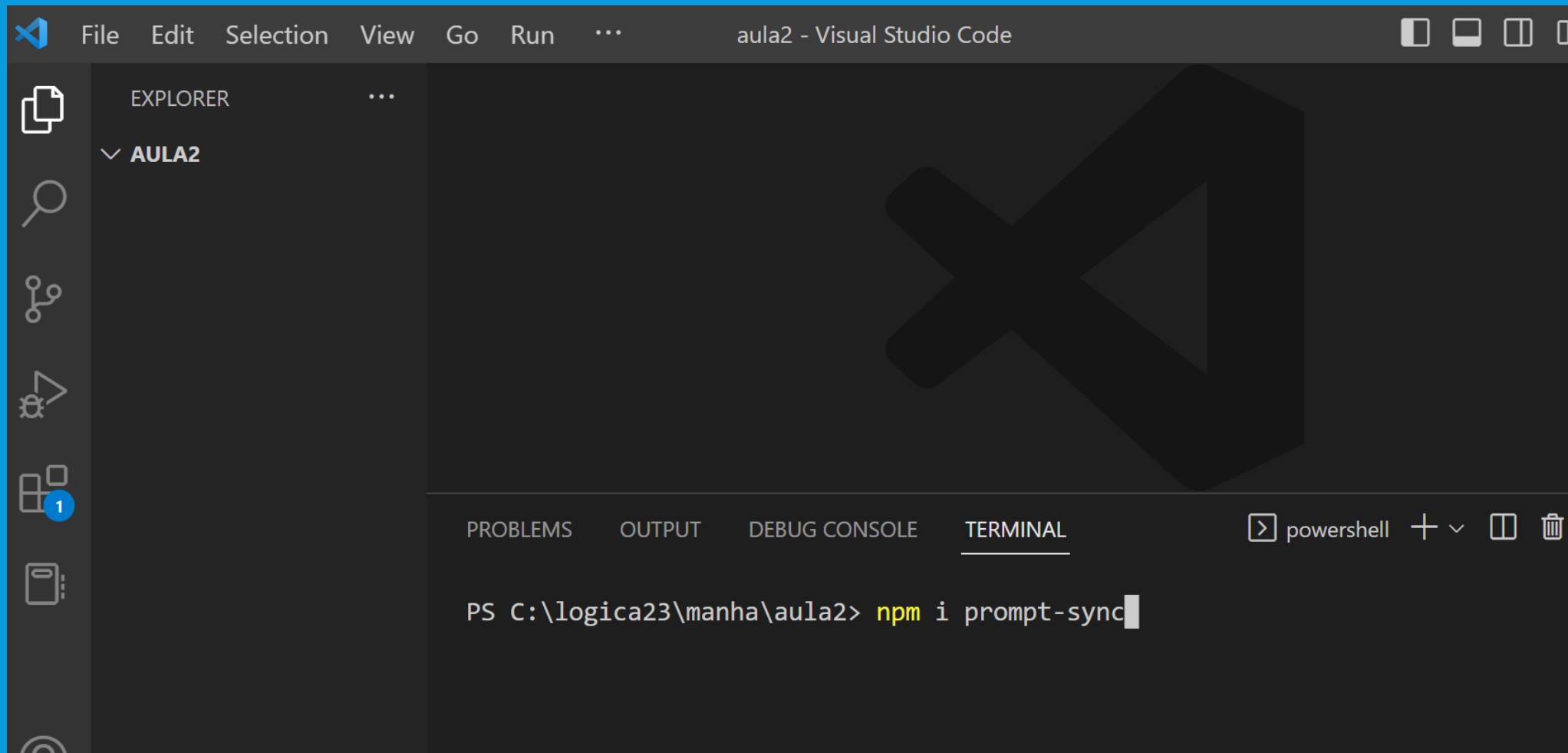


3. No VSCode, clique em Terminal / New Terminal

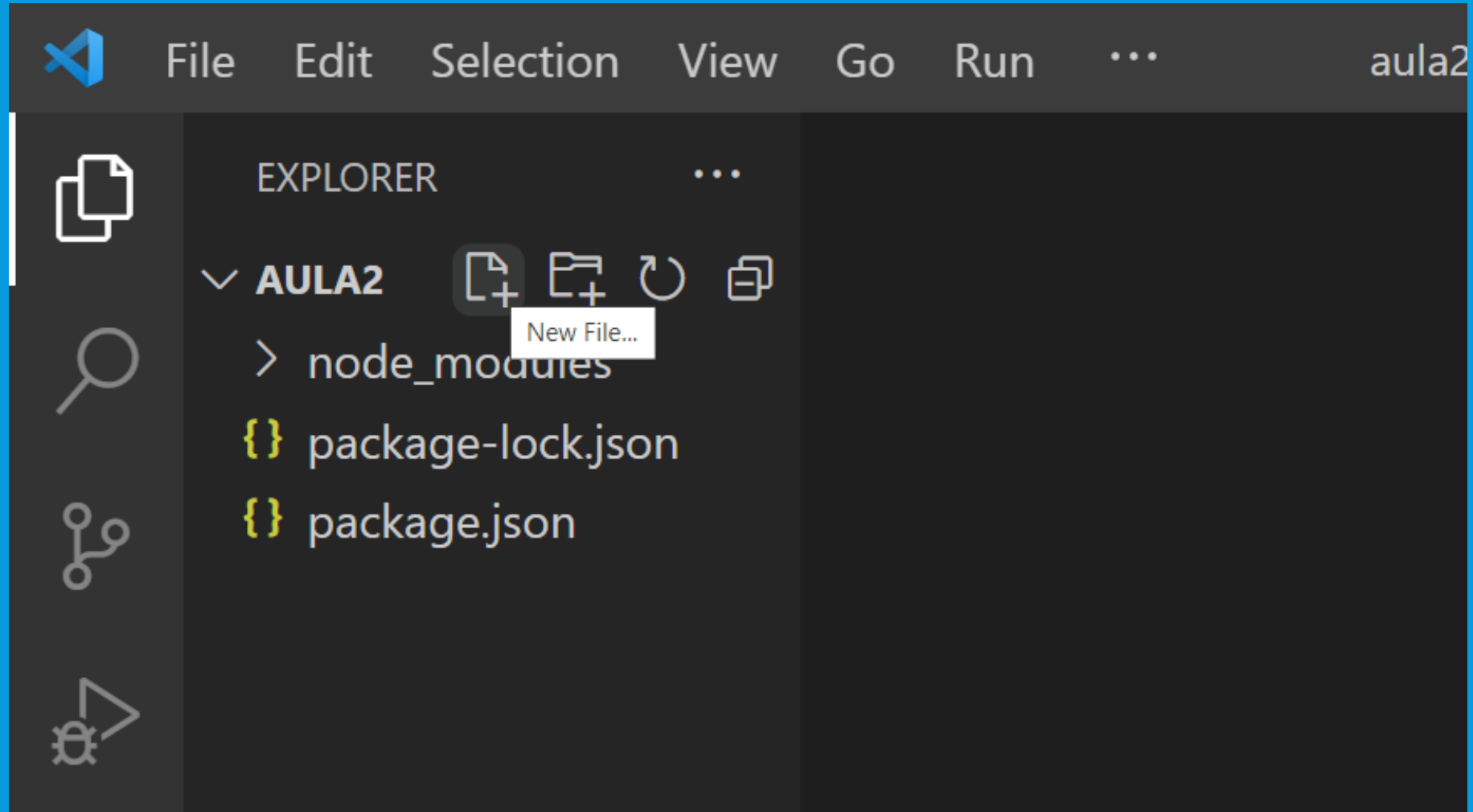


4. No terminal execute (apenas uma vez no diretório criado) o comando:

npm i prompt-sync

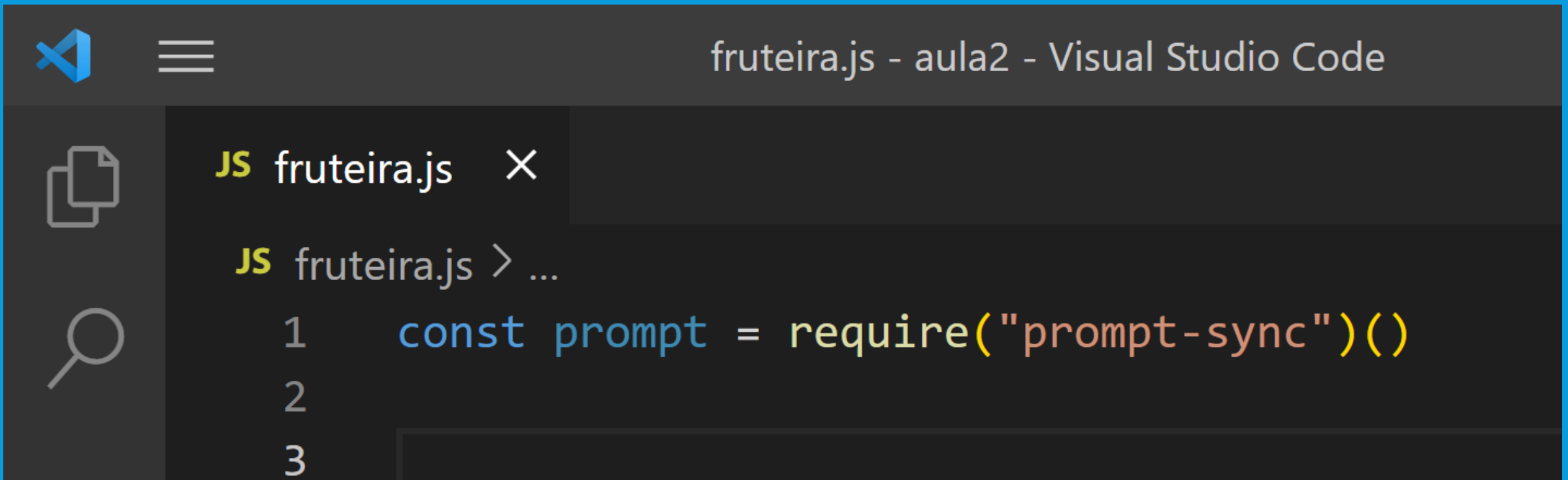


5. Feche o Terminal e crie um novo arquivo (Dê um nome para o arquivo com a extensão .js)



6. Insira a seguinte linha no início do programa

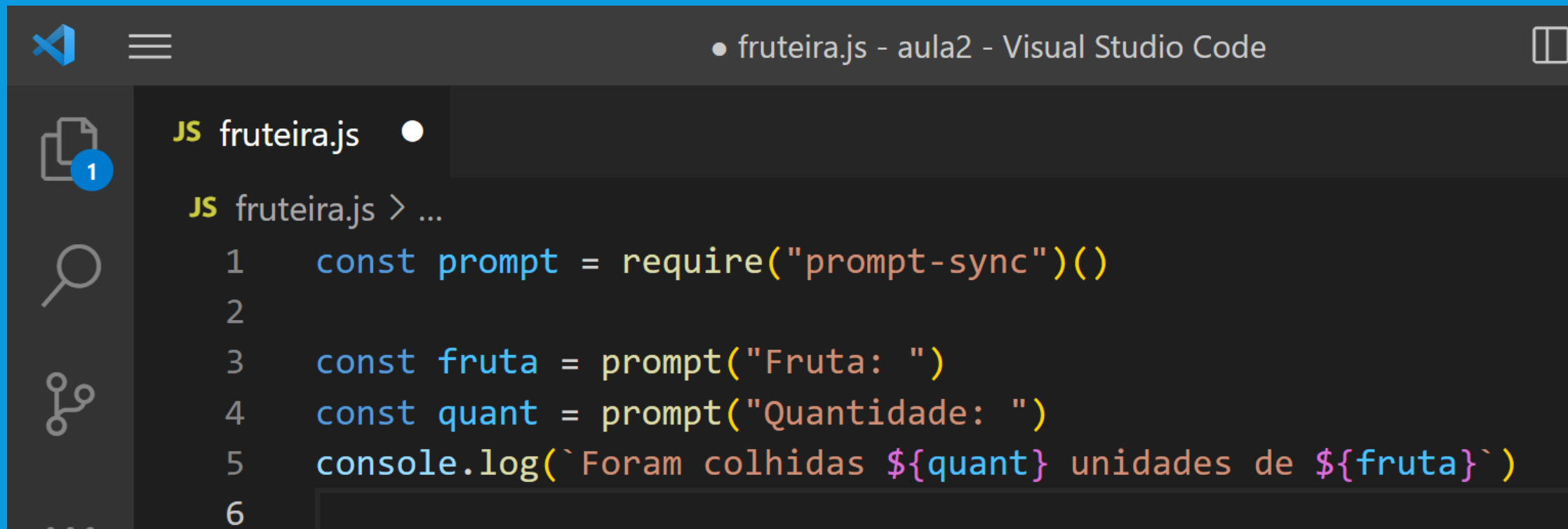
```
const prompt = require("prompt-sync")()
```



The screenshot shows the Visual Studio Code interface. The title bar at the top reads "fruteira.js - aula2 - Visual Studio Code". On the left sidebar, there is a file explorer icon, a search icon, and a tab labeled "JS fruteira.js" with a close button. The main editor area displays the following code:

```
JS fruteira.js > ...  
1  const prompt = require("prompt-sync")()  
2  
3
```

7. Insira as demais linhas do programa

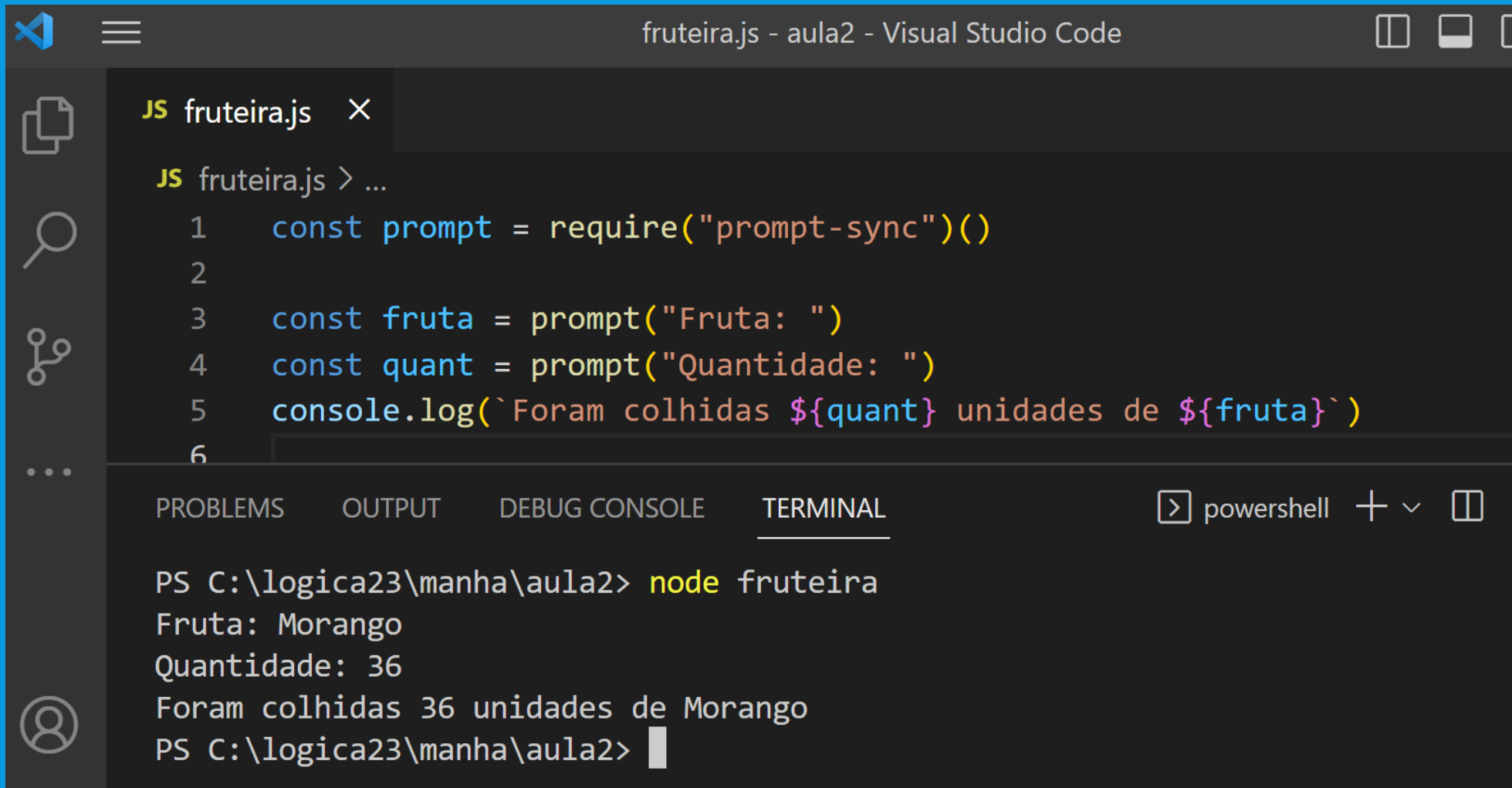


```
fruteira.js •  
JS fruteira.js > ...  
1  const prompt = require("prompt-sync")()  
2  
3  const fruta = prompt("Fruta: ")  
4  const quant = prompt("Quantidade: ")  
5  console.log(`Foram colhidas ${quant} unidades de ${fruta}`)  
6
```

8. Salve o arquivo (Control+S), abra o terminal e rode o comando

node fruteira

Ou seja, node e o nome do programa (não é necessário colocar a extensão)



The screenshot shows the Visual Studio Code interface. The top bar indicates the file is 'fruteira.js - aula2 - Visual Studio Code'. The editor displays the following JavaScript code in 'fruteira.js':

```
1  const prompt = require("prompt-sync")()
2
3  const fruta = prompt("Fruta: ")
4  const quant = prompt("Quantidade: ")
5  console.log(`Foram colhidas ${quant} unidades de ${fruta}`)
6
```

The bottom panel shows the 'TERMINAL' tab with a PowerShell session. The command 'node fruteira' has been executed, resulting in the following output:

```
PS C:\logica23\manha\aula2> node fruteira
Fruta: Morango
Quantidade: 36
Foram colhidas 36 unidades de Morango
PS C:\logica23\manha\aula2>
```

VARIÁVEIS

As variáveis servem para armazenar um valor em um programa. Devem ser identificadas por um nome.

As variáveis declaradas em um programa devem possuir um nome, seguindo algumas regras de nomenclatura. Em JavaScript, os nomes de variáveis não podem:

- Conter espaços.
- Começar por número.
- Conter caracteres especiais, como +, -, *, /, %, (,), {, }, !, @, #.
- Utilizar nomes de palavras reservadas da linguagem, como `function`, `var`, `new`, `for` ou `return`.

Variáveis escritas com letras maiúsculas são diferentes de variáveis escritas com letras minúsculas. O uso do caractere “_” é válido, porém dê preferência para declarar variáveis com o nome em letras minúsculas e o uso de uma letra maiúscula para destacar palavras compostas (padrão denominado *camelcase*). São exemplos de nomes válidos de variáveis: `cidade`, `nota1`, `primeiroCliente`, `novoSalario`, `precoFinal`, `dataVenda`. Procure usar nomes que indicam o conteúdo que a variável vai armazenar.

DECLARAR VARIÁVEIS

Em JavaScript, para declarar uma variável devemos utilizar os comandos:

- const
- let

Exemplos:

```
const nome = prompt("Nome: ")
```

```
const idade = prompt("Idade: ")
```

```
console.log(nome + ", sua idade é " + idade)
```

```
console.log(`${nome}, sua idade é ${idade}`)
```

PROCESSAMENTO

- O “processamento” em um programa, no geral, ocorre a partir da criação de novas variáveis.
- Estas variáveis realizam algum cálculo / processamento sobre os dados de entrada.
- E, as variáveis criadas no processamento, são utilizadas para compor a resposta do programa.

ATRIBUIÇÃO DE VALOR

Utilizamos o conceito de atribuição de valor para indicar que uma variável deve receber um determinado conteúdo (texto ou número, por exemplo).

Exemplo:

```
const num = prompt("Número: ")
```

```
const dobro = num * 2
```

```
console.log(`O dobro de ${num} é: ${dobro}`)
```

1.9 Tipos de dados e conversões de tipos

As variáveis manipuladas em um programa são de um determinado tipo. Em JavaScript, os tipos principais de dados são strings (variáveis de texto), números e valores booleanos (true ou false). Saber o tipo de uma variável nos permite identificar quais operações são possíveis para essa variável. Ou, então, qual o comportamento dessa variável nas fórmulas em que elas estão inseridas. Nesse contexto, há algumas particularidades na linguagem JavaScript. Vamos apresentar uma dessas particularidades no Exemplo 1.3, no qual o resultado do cálculo é exibido ao lado de cada variável como um comentário (//).

Exemplo 1.3 – Operações envolvendo strings e números (ex1_3.html)

```
<script>
  const a = "20"
  const b = a * 2    // b = 40
  const c = a / 2    // c = 10
  const d = a - 2    // d = 18
  const e = a + 2    // e = 202 ???
  alert("e: " + e)   // exibe o valor de uma variável
</script>
```

Nesse exemplo, temos uma variável do tipo string que recebe “20” (`const a = "20"`). Ela é entendida como sendo do tipo string por estar delimitada por aspas. Nas operações de multiplicação, divisão e subtração, a linguagem converte esse texto em número e o valor retornado está de acordo com o esperado. Contudo, quando realizamos a adição, o valor de retorno é diferente do padrão, pois a linguagem concatena (+) o texto com o número, algo semelhante ao que foi feito no Exemplo 1.2.

Função: Number()

- Podemos utilizar a função Number() para converter strings (textos) em números.

Exemplo:

```
const num = Number(prompt("Número: "))
```

```
const dobro = num + num
```

```
console.log(`O dobro de ${num} é: ${dobro}`)
```

EXERCÍCIOS

Elaborar os seguintes programas de entrada, processamento e saída.

- Elaborar um programa que leia 2 notas de um aluno. Calcule e mostre a média das notas.
- Elaborar um programa que leia o valor de um jantar. Sabendo que a taxa do garçom é de 10%, calcule e mostre a taxa do garçom e o valor total a ser pago.
- Elaborar um programa que leia o valor de um veículo. Mostre a promoção de financiamento da revenda, que consiste em 50% de entrada e o valor do saldo em 12X.
- Elaborar um programa que leia nome e idade de um aluno. Calcule o ano que ele nasceu. Exiba mensagem informando o nome do aluno e seu ano de nascimento.
- Elaborar um programa que leia um número. Exiba os vizinhos deste número (anterior e posterior)